(2)

SWITCH: A SIMULATION OF REPRESENTATIONAL CHANGE IN THE MUTILATED CHECKERBOARD PROBLEM

Technical Report AIP - 108

Craig A. Kaplan

Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213

December 8, 1989

The Artificial Intelligence and Psychology Project

Departments of
Computer Science and Psychology
Carnegie Mellon University

Learning Research and Development Center
University of Pittsburgh

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; Distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AIP - 108	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Carnegie Mellon University 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Computer Sciences Division Office of Naval Research (Code 1133)
6c. ADDRESS (City, State, and ZIP Code) Department of Psychology Pittsburgh, PA 15213	7b. ADDRESS (City, State, and ZIP Code) 800 N. Quincy Street Arlington, VA 22217-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Same as Monitoring Organization 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-86-K-0678
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS p40005ub201/7-4-86

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
N/A	N/A	N/A	N/A

11. TITLE (Include Security Classification)
Switch: A simulation of representation change in the mutilated checkerboard problem

12. PERSONAL AUTHOR(S) Craig A. Kaplan

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM 86Sept15 TO 91Sept14	89/12/8	25

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

One of the best ways to study insight is to study problems that require shifts in representation. One such problem is The Mutilated Checkerboard (MC) problem. This report presents a brief psychological account of problem solving in the MC domain, followed by a detailed computer simulation of how change of representation might actually occur. The computer simulation, SWITCH, was build in the production system language, Soar. Analysis of how SWITCH works leads to psychological claims about problem solving strategies, representation, and the mechanism underlying representational change.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED [✓ SAME AS RPT.] ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Alan L. Meyrowitz	22b. TELEPHONE (Include Area Code) (202) 696-4302 22c. OFFICE SYMBOL N00014

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

2

SWITCH: A SIMULATION OF REPRESENTATIONAL CHANGE IN THE MUTILATED CHECKERBOARD PROBLEM

Technical Report AIP - 108

Craig A. Kaplan

Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213

December 8, 1989

This research was supported in part by the Computer Sciences Division, Office of Naval Research, under contract number N00014-86-K-0678, and in part by the Department of Psychology at CMU. Support for final revisions of the report came from IBM's Human Factors Center at Santa Teresa Laboratories. Reproduction in whole or part is permitted for any purpose of the United States Government. Approved for public release; distribution unlimited.

The author is indebted to Herb Simon, Davide Steier and Allen Newell for their substantive help with this project. The author also wishes to thank Elaine Atkinson for her help in formatting the final copy.

Correspondence and requests for reprints should be sent to Herbert A. Simon, Department of Psychology, Carnegie-Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213

INTRODUCTION

Insight is that flash of illumination during which a problem solver exclaims "Aha!" and sees (or thinks s/he see) the answer. To have an insight, one must first have a problem that constitutes something of a puzzle. If the problem is difficult for merely technical reasons (e.g. the following of a long and tedious algorithm), there is little room for insight. However, if the problem requires that one or more of its elements be thought of in a new way, then insight seems quite likely.

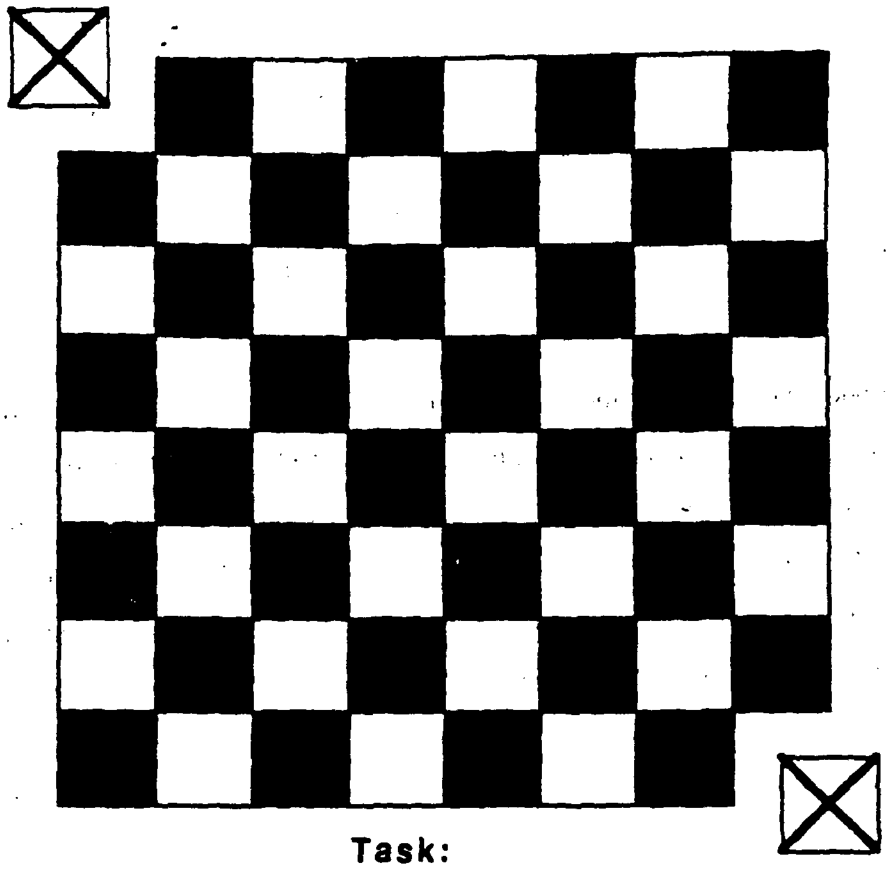
There is good evidence that insight often follows rapidly after the problem solver represents the problem in a new way (Kaplan & Simon in press). Unfortunately, it quite difficult to collect psychological data at the exact moment of insight, since human subjects are almost universally silent just before the AHA!. This report presents a computer simulation of how subjects might change representation and how that representation might lead to the solution of an infamous insight problem, the Mutilated Checkerboard problem.

The Mutilated Checkerboard Problem

The Mutilated Checkerboard problem (McCarthy 1964, Newell 1966, Kaplan and Simon in press) is an ideal problem for the study of insight because its solution requires a radical shift in representation. The problem it has been presented it in a number of experiments follows:

> Subjects are presented with an 8x8 checkerboard from which the diagonally opposite corners have been removed (see Figure 1). Subjects are asked to imagine that they have 31 dominos, each of which is capable of covering two squares if it is placed on the checkerboard either horizontally or vertically. Diagonal placement is not allowed. The task is to determine if it is possible to cover the 62 remaining squares using the 31 dominos. A covering must be shown, or the subject must logically prove why a covering is impossible to produce.

In fact no covering exists. An insightful way to prove this fact is to notice that a domino must cover a black and a white square no matter how it is placed on the board. Since all 31 dominos must be used if an area of 62 squares is to be covered, there must be equal numbers of black and white squares (each pair corresponding to one domino placement) for the problem to be possible. However, an examination of Figure 1 reveals that in removing the diagonally opposite corners, we have removed two white squares. That means there are two more blacks than whites left on the board, and the problem is impossible to solve.



Task:

Cover the 62 remaining squares using 31 dominos. Each domino covers two adjacent squares. **Or:** Prove logically why such a covering is impossible.

Figure 1: The Classic Mutilated Checkerboard (MC) Problem

Elsewhere the interesting characteristics of the MC problem and of solution attempts by human subjects are described and analyzed in considerable detail (Kaplan and Simon in press). However in order to establish a psychological context for the simulation which follows, I will re-iterate some of those characteristics here.

First, the switch in representation from the generic concept of "square" to the elaborated concepts of "black square" and "white square" is at the heart of the Mutilated Checkerboard's difficulty. More specifically, if we distinguish between time spent exploring various approaches prior to switching representation, and time spent on the problem after the switch, it becomes clear that most of the problem's difficulty stems from intial exploration of fruitless paths prior to the switch. Once the switch has been made, the proof is trivial for many subjects.

The switch in representation itself occurs quite rapidly in what one subject retrospectively called "a flash of insight." Figure 2 presents the protocol transcripts from three of the more articulate subjects before, during, and after representational change.[1] Notice that the total episode -- from receiving the hint or first becoming aware of the idea of parity (e.g. the alternating color pattern), through the actual shift in representation (which is not always clearly delineated), to the generation of a rough proof of impossibility -- is fairly brief. Typically, it takes less than one minute. This short time span stands in stark contrast to the 20 to 45 minutes that subjects typically spend exploring fruitless paths (even though they are given periodic hints to prod them along!).

[1]These protocols are taken from a study (Kaplan and Simon in press) in which several different versions of the Mutilated Checkerboard problem were used. The versions differed in that sometimes the squares were labelled with the alternating words such as "bread" and "butter" instead of actually being different colors. However, the reader should feel free to translate "bread" and "butter" as "black" and "pink."

SUBJECT 1 (a BREAD & BUTTER subject): EXCERPT LASTS: 70 Secs.

1: Just by trial and error I can only find 31 places ... I dunno, maybe someone else would have counted the spaces and just said that you could fit 31; but if you try it out on the paper, you can only fit 30. (pause & distracted chattering)

E: Keep trying.

1: Maybe it has to do with the words on the page? I haven't tried anything with that. (pause)
Maybe that's it. Ok, dominos, umm, the dominos can only fit ... alright, the dominos can fit over two squares, and no matter which way you put it because it cannot go diagonally, it has to fit over a butter and a bread. And because you crossed out two breads, it has to leave two butters left over so it doesn't ... only 30, it won't fit. Is that the answer?

SUBJECT 2 (a COLOR subject): EXCERPT LASTS: 48 Secs.

2: There's an even number of squares, so it's possible depending on the placement... so it has to be the placement... (pause) ...

2: How about a different placement? We could try that.
Well, if we place the Xs in different corners, then it'd be really simple ... other than opposite .... ummmm ....How about a black and a pink ..... Oh, we always have to cover a black and a pink square... at the same time time .... Uh, there's no way to avoid that ... ummm.

Oh!, There's two black squares covered up and ... since you always have to cover up a black and a pink square, there's no way you can do it.

SUBJECT 7 (a COLOR subject requiring a hint): EXCERPT LASTS: 36 Secs.

E: What about the color? Can you use color to help you out?

7: There's two pinks next to each other .... Oh God!! You're taking two black out? And you would need to take a black and a white out ... a black and a pink out. (pause)

7: So you're leaving ... OH!!! Jeez! So you're leaving .... it's short -- how many, you're leaving uhhhh .... there's more pinks than black, and in order to complete it you'd have to connect two pinks but you can't because they are diagonally ... is that getting close? ... since they are diagonally connected ... and so you're always gonna end up with two extra pinks ... because their mates were taken out.

Figure 2: The AHA! Experience (3 Protocol Excerpts)

I suggest that the rapid generation of the rough proof once subjects switch to the appropriate representation corresponds to a relatively straightforward application of fairly general knowledge. On the other hand, the initial (and time consuming) problem solving can be interpreted as search through a large space of potential cues without powerful heuristics to narrow that search. How subjects eventually arrive at cues which trigger a switch in representation is discussed elsewhere (Kaplan and Simon in press). The purpose of the production system simulation, SWITCH, is to demonstrate a set of mechanisms sufficient to explain the actual switch of representation itself, and to explain how this switch might lead to a rapid solution. SWITCH aims to illuminate those processes that are hidden in the pauses and unspoken words of the brief verbal protocols in Figure 2.

The SWITCH Simulation

Before plunging ahead with the actual performance of SWITCH, we must establish some basic facts about Soar -- the particular production system chosen for implementing SWITCH. We also should attend to the features of Soar that have psychological meaning in SWITCH, and to the knowledge that SWITCH starts with.

Soar Basics

SWITCH makes primary use only of the fact that Soar is a production system (i.e. it supports a set of rules that match against the contents of WM to see if they are instantiated.) However, Soar is also an architecture for general intelligence possessed of a number of special features that make it unique among production systems (Laird, Rosenbloom, & Newell 1986).[2]

Soar's claims that all cognition takes place in problem spaces and that all learning occurs by chunking would be critical if SWITCH was expanded to model the entire course of solving the MC problem, as opposed to focussing on the moment of insight. The psychological claims that are being made at present however, include only that: 1) The knowledge that subjects have stored in LTM can be represented by productions, 2) that the condition side of the productions specify what retrieval cues might access that knowledge, 3) and that the contents of WM correspond roughly to the contents of the subjects' STM

[2]A non-exhaustive list of these features includes the following: A two-phase processing cycle consisting of an elaboration phase during which productions fire "in parallel" and a decision phase during which a new goal, problem space, or operator is made part of the currently active context, A highly specified learning method -- chunking, An architecture centered around the notion of problem spaces, & goal generation driven by impasses reached during problem solving.

combined with the information that is perceptually available in the environment.

What SWITCH Starts With

SWITCH starts with essentially the same information as a subject who has already done a significant amount of unsuccessful problem solving and has just been given a hint to pay attention to the color of squares. In addition to modelling the behavior of the subject, however, SWITCH has the task of modelling the environment in which the subject acts. These two sources of knowledge -- knowledge about the task environment, and about the subject's representation of that environment -- have been carefully distinguished and separated in the SWITCH. Specifically, SWITCH is given the following information at the start of a simulation run:[3].

- A model of the real world problem (e.g. representations of squares, dominos, the adjacency relationships between squares)
- A model of the human subject's representation of the real world problem, including concepts that have been generated during problem solving, prior to receiving the color hint (e.g. a concept of a generic square, the proposition that a domino covers two squares)
- An assumed focus of attention (i.e. a 2x2 patch of the board that is referred to first when the simulation needs information about real world squares)
- A set of fairly general productions corresponding to well learned inference rules presumably possessed by adult subjects (e.g. if one proposition appears true based on observation and the same proposition seems false logically, then a contradiction exists).
- Strategic knowledge (implemented in domain specific productions for the purpose of this version of the simulation) corresponding to general strategies such as: "pursue hot ideas" or "change to finer grain size upon failure."
- A hint (corresponding to that given to subjects) that the parity (e.g. color) of the squares is important.

How it works

SWITCH, has three distinct levels of representation. At bottom are the real world elements (RWEs) -- the simulation's model of the problem elements themselves. Next comes the Internal Representational Concept (IRC) level which corresponds to the subject's internal representation of elements in the external world. Finally, there is the propositional level which corresponds to sequences of IRCs strung together.

While the RWEs are necessary in that the simulation must model the task domain, the way these units are represented in SWITCH is irrelevant to the subject's internal representation of the problem. Hence there is no psychological claim at the RWE level.

[3]Note: The complete Soar code for SWITCH, along with a sample execution trace can be found in Appendix A

In co..trast, the IRCs and Propositions correspond to a human subject's representation of the problem. The main psychological claim here is that the representation is hierarchical in nature. One might think of the IRCs as representational primitives which are combined in different ways to produce various propositions.

SWITCH's hierarchial representation of knowledge provides it with two basic methods of solving problems: 1) it can try to produce new combinations of the primitives it already has in the hopes that the new propositions will trigger some useful knowledge that it has already learned, or 2) it can try to elaborate the IRCs in the hopes that changing the building blocks themselves will eventually result in useful propositions.

The best way to get a feel for how the simulation works is to examine a production and see what it does. Figure 3 (below) presents the production that performs the actual shift in representation. This production matches on a hint and then checks to see if any relevant IRCs exist that are unelaborated with respect to the attribute that the hint refers to. Thus, if the hint says to attend to the color of the squares, the production may find an IRC corresponding to a generic square -- that is to the concept of "squareness" without any value for color. At this point, the production looks at the board (the RWEs) to see if the squares in the real world have color. They do, so the production maps the color from the real world square to a new IRC that possesses all the previous attributes of the generic square, but now also specifies the square's color. Thus, by "analogy" to the real world, the simulation is able to shift from an intial representation of "square", to a representation of "black square" or "white square." A similar production allows the simulation to elaborate old propositions using new IRCs. Thus the proposition "A domino covers a square and a square" may become "A domino covers a black square and a white square."[4]

[4]The details of this production, named Elaborate-propositions-by-analogy, can be found in Appendix A. Note that knowledge in this simulation is monotonic so the old proposition is not actually transformed, but rather a new proposition is created using the old proposition as a template from which to "analogize."

```
Production: elaborate-concept-by-analogy

IF:  The goal is to prove the problem impossible, AND
     The operator is to elaborate a representation, AND
     A hint exists saying pay attention to some attribute (e.g. COLOR), AND
     Some representational concepts (e.g. the concept of squares) exist
     that have no value for the attribute in question (e.g. COLOR), AND
     There are some real world referents for the representational concepts
     that can be referred to (e.g. the squares which really exist
     on the board)

THEN: Map the value of the hinted-at attribute (e.g. COLOR) from the real
      world objects (e.g. real squares) to the representational concept of the
      objects (e.g. representation of squares).
```

Figure 3: A Sample Production From SWITCH

What it does

An actual trace of the simulation run can be found in Appendix A, but I have abstracted the main sequence of steps and present it below:

1) Get the hint.

2) Decide to elaborate IRCs. (This strategic decision reflects the fact that the simulation has been stuck up to this point, and examines the representational primitives since no progress has been made at the higher propositional level).

3) Elaborate concepts by analogy. (The simulation comes up with the new IRCs of "black square" and "white square").

4) Decide to generate propositions. (Once new IRCs have been generated, the strategy of "pursue hot ideas" dictates that the simulation check what the implications of the new conceptual primitives will be at the propositional level).

5) Elaborate propositions by analogy. (The simulation produces the proposition that a domino covers a "black square" and a "white square").

6) Infer equal numbers covered. (The newly generated proposition -- step 5 -- triggers knowledge that equal numbers of the two types of squares must be covered. Pilot data indicates that subjects have a production similar to this in general form).

7) Check actual numbers covered. (Since the simulation is working within the general context of the schema "Proof by contradiction," every new fact must be checked against reality)

8) Detect a contradiction and exclaim "Impossible!"

9) Decide to generate a reason for impossibility. (Again, the proof context dictates that the simulation search for a reason for the contradiction)

10) Trace back from contradiction. (The simulation has stored the source of its proposition -- logically deduced, or empirically observed -- and is able to recall them).

11) State rough proof. (The simulation uses general knowledge about proofs to frame the information it has recalled).

The behavior outlined above captures very well the behavior of some subjects from the time they receive a hint to the time they generate a rough proof, however the majority of subjects deviated in various ways from the account just presented. Since an examination of the protocols reveals that the seeds of these deviations can often be traced to behavior during earlier problem solving, many individual differences could probably be captured in a simulation of the entire problem solving episode. With regard to change of representation, however, subjects seem remarkably consistent in showing surprise, and then rapid utilization of the new representation. Thus, although there are individual differences in the paths that subjects take once they have changed representations, there is no need to suppose that there is any variability in the mechanism underlying the representational shift itself.

Conclusions

SWITCH makes a number of psychological claims. First, there is a general correspondences between the production system architecture and the psychological notion of retrieval from LTM. Moreover, the information "built in" to the simulation corresponds the information that human subjects could reasonably be expected to have.

Second, the hierarchical representation scheme (e.g. the levels of RWEs, IRCs, and propositions) seems psychologically valid. That is, it seems reasonable that representations are build of more finely grained representational units from the level below. However, it remains to be seen where the boundaries to these levels might be, and to what degree changes at one level are likely to affect the representations at another.

Third, the simulation has incorporated the heuristics of "change grain size upon failure" and "pursue hot ideas." While there is some indication that subjects use these strategies in the Mutilated checkerboard problem, the generality of these heuristics needs to be tested further.

Fourth, the failure of the simulation to match the behavior of all of the subjects emphasizes the importance of individual differences in problem solving. Many of these individual differences could probably be captured in a simulation of the entire problem solving episode (including the many false starts that typically precede insight).

Finally, the simulation provides a mechanism for changing representations (analogical mapping), and illustrates how general knowledge, together with a hint, is sufficient to produce the phenomenon of insight in the MC problem.

REFERENCES

Kaplan, C.A, and Simon, H.A. (in press) In search of insight. *Cognitive Psychology.*

Laird, J.E., Newell, A., & Rosenbloom, P. S. (1986) Soar: An architecture for general intelligence. *Computer Science Technical Report # CMU-CS-86-171* Pittsburgh: CMU.

McCarthy, J. (1964) A tough nut for proof procedures. *Stanford Artifical Intelligence Project*, Memo No. 16.

Newell, A. (1966) On the representations of problems. *Computer Science Research Review*, 18-33. Pittsburgh: CMU.

Simon, H.A. (1978) On the forms of mental representation. In C. Wade Savage (ed.) *Perception and Cognition: Issues in the Foundation of Psychology*, Vol. IX, Minnesota Studies in the Philosophy of Science. Minneapolis, U. of Minnesota Press.

APPENDIX A: -- The Soar Simulation:

Trace and Documented Soar Code

```
0   g: g00005
1:1 load-top-goal
1:2 DECIDE problem-space p00006
1   p: p00006 proof-by-contradiction
2:3 load-problem-features
2:4 DECIDE state s00007
2   s: s00007
3:5 create-generate-proposition
3:5 create-elaborate-representation
3:5 load-initial-propositions
3:5 load-initial-concepts
3:5 load-hints
3:6 default-prefer-elaborate-representation
3:7 DECIDE operator o000049
3   o: o000049 elaborate-representation
4:8 elaborate-concept-by-analogy
4:8 elaborate-concept-by-analogy
4:9 newstate*set-up-state-for-copying
4:10 newstate*copy-valid-state-attributes
4:10 newstate*copy-valid-state-attributes
4:10 newstate*copy-valid-state-attributes
4:11 DECIDE state n00064
4   s: n00064
5:12 default*no-operator-retry
5:12 create-generate-proposition
5:12 create-elaborate-representation
5:13 prefer-generate-proposition1
5:14 DECIDE operator o00065
5   o: o00065 generate-proposition
6:15 elaborate-propositions-by-analogy
6:16 newstate*set-up-state-for-copying
6:16 infer-equal-numbers-covered
6:17 newstate*copy-valid-state-attributes
6:17 newstate*copy-valid-state-attributes
6:17 newstate*copy-valid-state-attributes
6:17 newstate*copy-valid-state-attributes
6:17 newstate*copy-valid-state-attributes
6:17 make-count-proposition
6:18 contradiction-found
 The problem is impossible!
6:19 DECIDE state n00068
6   s: n00068
7:20 default*no-operator-retry
7:20 create-generate-proposition
7:20 create-elaborate-representation
7:20 create-find-reason
7:21 prefer-generate-proposition2
7:21 prefer-generate-proposition1
7:21 prefer-find-reason
7:22 DECIDE operator o00072
7   o: o00072 find-reason
8:23 trace-back-contradiction
For the problem to be possible, it must be true that
number black square equal number white square
since domino covers black square and white square .
But, it is false that
number black square equal number white square
by empirical observation.  Therefore, the
problem is impossible.
"End -- Explicit Halt"
```

```
;*******************************************************************
;*    Mutilated Checkerboard Problem -- Simulation in SOAR 4.4      *
;*                                                                  *
;*                                                                  *
;*                      Craig Kaplan                                *
;*                      Carnegie-Mellon University                  *
;*                      MARCH 9, 1987                               *
;*                                                                  *
;* This program simulates the behavior of subjects from the time    *
;* that they receive the COLOR hint to the time that they generate  *
;* a rough proof of the problem's impossibility.                    *
;********************************************************************
;
;============================================================================
;       INTIALIZATION PRODUCTIONS

(sp  load-top-goal
;------------------
;; Establish the top level goal of proving the problem impossible as well
;; as the method of proof by contradiction.  The name of the problem-
;; space is somewhat arbitrary since its states really represent the
;; ever-changing WM-representation.

        (goal <g> ^problem-space undecided  - ^supergoal)
-->
        (goal <g> ^name prove-impossible)
        (problem-space <p> ^name proof-by-contradiction)
        (preference   <p> ^role problem-space ^value acceptable ^goal <g>))
)

(sp load-problem-features
;-------------------------
;; Load a set of Real World Elements (RWEs)
;; corresponding to the physical squares, dominos, and adjacent-squares.
;; These elements correspond to the problem-itself as opposed to a human's
;; representation of the problem.

        (goal <g> ^problem-space <p> ^state undecided - ^supergoal)
        (problem-space <p> ^name proof-by-contradiction)
-->
        (preference <s> ^role state ^value acceptable ^problem-space <p>
                ^state undecided)
        (state <s> ^rw <rw> ^ks <ks>)
        (rwe <rw1> ^name domino ^shape rectangle ^area 2 ^number-of 1
                 ^function coverer)
        (rwe <rw2> ^name square ^shape square ^area 1 ^number-of 1 ^position 1
                 ^color-of black ^function coveree ^status removed)
        (rwe <rw3> ^name square ^shape square ^area 1 ^number-of 1 ^position 2
                 ^color-of white ^function coveree ^status present)
        (rwe <rw4> ^name square ^shape square ^area 1 ^number-of 1 ^position 3
                 ^color-of black ^function coveree ^status present)
        (rwe <rw5> ^name square ^shape square ^area 1 ^number-of 1 ^position 4
                 ^color-of white ^function coveree ^status present)
        (rwe <rw6> ^name square ^shape square ^area 1 ^number-of 1 ^position 5
                 ^color-of black ^function coveree ^status present)
        (rwe <rw7> ^name square ^shape square ^area 1 ^number-of 1 ^position 6
                 ^color-of white ^function coveree ^status present)
        (rwe <rw8> ^name square ^shape square ^area 1 ^number-of 1 ^position 7
         ^color-of black ^function coveree ^status present)
(rwe <rw9> ^name square ^shape square ^area 1 ^number-of 1 ^position 8
         ^color-of white ^function coveree ^status present)
(rwe <rw10> ^name square ^shape square ^area 1 ^number-of 1 ^position
         9 ^color-of black ^function coveree ^status present)
(rwe <rw11> ^name square ^shape square ^area 1 ^number-of 1 ^position
         10 ^color-of white ^function coveree ^status present)
(rwe <rw12> ^name square ^shape square ^area 1 ^number-of 1 ^position
         11 ^color-of black ^function coveree ^status present)
(rwe <rw13> ^name square ^shape square ^area 1 ^number-of 1 ^position
         12 ^color-of white ^function coveree ^status present)
(rwe <rw14> ^name square ^shape square ^area 1 ^number-of 1 ^position
         13 ^color-of black ^function coveree ^status removed)
(rwe <rw15> ^name square ^shape square ^area 1 ^number-of 1 ^position
         14 ^color-of white ^function coveree ^status present)
(rwe <rw16> ^name square ^shape square ^area 1 ^number-of 1 ^position
         15 ^color-of black ^function coveree ^status present)
(rwe <rw17> ^name square ^shape square ^area 1 ^number-of 1 ^position
         16 ^color-of white ^function coveree ^status present)
(rwe <rw19> ^name adjacent-squares ^shape square ^number-of 2 ^pos1 2
         ^pos2 3)
(rwe <rw20> ^name adjacent-squares ^shape square ^number-of 2 ^pos1 2
         ^pos2 7)
(rwe <rw21> ^name adjacent-squares ^shape square ^number-of 2 ^pos1 7
         ^pos2 8)
(rwe <rw22> ^name adjacent-squares ^shape square ^number-of 2 ^pos1 8
         ^pos2 9)
(rwe <rw23> ^name adjacent-squares ^shape square ^number-of 2 ^pos1 9
         ^pos2 10)
(rwe <rw24> ^name adjacent-squares ^shape square ^number-of 2 ^pos1 9
         ^pos2 16)
(rwe <rw25> ^name adjacent-squares ^shape square ^number-of 2 ^pos1 16
         ^pos2 15)
(rwe <rw26> ^name adjacent-squares ^shape square ^number-of 2 ^pos1 10
         ^pos2 15)
(rwe <rw27> ^name adjacent-squares ^shape square ^number-of 2 ^pos1 7
         ^pos2 10)
(rwe <rw28> ^name adjacent-squares ^shape square ^number-of 2 ^pos1 3
         ^pos2 6)
(rwe <rw29> ^name adjacent-squares ^shape square ^number-of 2 ^pos1 6
         ^pos2 7)
(rwe <rw30> ^name adjacent-squares ^shape square ^number-of 2 ^pos1 6
         ^pos2 11)
(rwe <rw31> ^name adjacent-squares ^shape square ^number-of 2 ^pos1 10
         ^pos2 11)
(rwe <rw32> ^name adjacent-squares ^shape square ^number-of 2 ^pos1 15
         ^pos2 14)
(rwe <rw33> ^name adjacent-squares ^shape square ^number-of 2 ^pos1 11
         ^pos2 14)
(rwe <rw34> ^name adjacent-squares ^shape square ^number-of 2 ^pos1 11
         ^pos2 12)
(rwe <rw35> ^name adjacent-squares ^shape square ^number-of 2 ^pos1 5
         ^pos2 12)
(rwe <rw36> ^name adjacent-squares ^shape square ^number-of 2 ^pos1 5
         ^pos2 6)
(rwe <rw37> ^name adjacent-squares ^shape square ^number-of 2 ^pos1 3
         ^pos2 4)
(rwe <rw38> ^name adjacent-squares ^shape square ^number-of 2 ^pos1 4
         ^pos2 5)
(rv <rv> ^rwe <rw1> <rw2> <rw3> <rw4> <rw5> <rw6> <rw7> <rw8> <rw9>
                  <rw10> <rw11> <rw12> <rw13> <rw14> <rw15> <rw16> <rw17>
                  <rw18> <rw19> <rw20> <rw21> <rw22> <rw23> <rw24> <rw25>
                  <rw26> <rw27> <rw28> <rw29> <rw30> <rw31> <rw32> <rw33>
                  <rw34> <rw35> <rw36> <rw37> <rw38>)
)

(sp load-hints
;--------------
;; Load the "color-hint" to pay attention to the color of the squares,
;; and the "insight-hint" which alerts the Ss that there is a "trick
;; way of looking at the problem."

        (goal <g> ^problem-space <p> ^state <s> ^operator undecided)
        (problem-space <p> ^name proof-by-contradiction)
        (state <s> ^ks <ks>)
        (ks <ks> -^hint)
-->

        (hint <h1> ^name color-of-squares-hint ^attend-to-name square
                  ^attend-to-attribute color-of)
        (hint <h2> ^name insight-hint)
        (ks <ks> ^hint <h1> <h2>)
)

(sp load-initial-concepts
;-------------------------
;; Create Internal Representational Concepts (IRCs) corresponding to the
;; concepts a subject is likely to have acquired during the course of
;; problem solving through the point when the subject gets the COLOR hint.
;; The concept "sample" represents the fact that subjects usually choose
;; to attend to a couple of specific squares rather than the entire board
;; at once.  Future simulations may model the sampling processes dynamically.
;; NOTE: At a later stage, the simulation will actually derive these IRCs
;; from RWEs using Concept Formation Rules. (Also, including "number black
;; square" and "number white square" here is a temporary kludge, as it is
;; almost certain that these concepts would be induced AFTER, not before
;; the COLOR hint.  We need to find an elegant way for SOAR to count squares.)

        (goal <g> ^problem-space <p> ^state <s> ^operator undecided)
        (problem-space <p> ^name proof-by-contradiction)
        (state <s> ^ks <ks>)
        (ks <ks> -^irc)

-->
        (irc <ir0> ^name square concept ^shape square ^area 1 ^number-of 1
                 ^position nil ^color-of nil ^function coveree ^status nil)
        (irc <ir1> ^name adjacent-squares concept ^shape square square ^number-of 2
                 ^area 2 ^pos1 3 ^pos2 4 ^function coveree )
        (irc <ir2> ^name domino concept ^shape rectangle ^area 2
                 ^function coverer )
        (irc <ir3> ^name number black square ^number 6 )
        (irc <ir4> ^name number white square ^number 8 )
        (irc <ir6> ^name sample ^shape square ^number-of 1 ^position 3
                 ^adjacent-to 4 ^salience 1st )
        (irc <ir7> ^name sample ^shape square ^number-of 1 ^position 4
                 ^adjacent-to 3 ^salience 2nd )
        (sf <sf1> ^name salient-feature ^attend-to-name square
                 ^attend-to-attribute number-of )
        (ks <ks> ^irc <ir0> <ir1> <ir2> <ir3> <ir4> <ir5> <ir6> <ir7>
                 ^sf <sf1>)
```

```
)

(sp load-initial-propositions
;-----------------------------
;; Loads propositions that would have been inferred by subjects prior to
;; receiving the COLOR hint. Note: This is a minimal, not exclusive list.

        (goal <g> ^problem-space <p> ^state <s> ^operator undecided)
        (problem-space <p> ^name proof-by-contradiction)
        (state <s> ^ks <ks>)
        (ks <ks> -^prop)
-->
        (ks <ks> ^prop <pr1>)
        (prop <pr1> ^term1 domino ^rel covers ^term2 square ^term3 square)
)

;=============================================================================
;        SEARCH CONTROL PRODUCTIONS -- PROOF-BY-CONTRADICTION PROBLEM SPACE


(sp  create-elaborate-representation
;-------------------------------------------
;; Create one of the operators that we assume subjects have whenever
;; they have had difficulty solving problems with the intial representation.

        (goal <g> ^name prove-impossible ^problem-space <p> ^state <s>
                ^operator undecided)
        (problem-space <p> ^name proof-by-contradiction)
        (state <s> ^rw <rw> ^ks <ks>)
-->
        (operator <o> ^name elaborate-representation)
        (preference <o> ^role operator ^value acceptable ^goal <g>
         ^problem-space <p> ^state <s>)
)

(sp  create-generate-proposition
;-------------------------------------------
;; Create one of the operators that we assume subjects have whenever
;; they are reasoning with propositions.

        (goal <g> ^name prove-impossible ^problem-space <p> ^state <s>
                ^operator undecided)
        (problem-space <p> ^name proof-by-contradiction)
        (state <s> ^rw <rw> ^ks <ks>)
-->
        (operator <o> ^name generate-proposition)
        (preference <o> ^role operator ^value acceptable ^goal <g>
         ^problem-space <p> ^state <s>)
)

(sp default-prefer-elaborate-representation
;------------------------------------------
;; If no new elaborations or propositions have
;; been generated, and the insight hint has been given,
;; then try to elaborate the basic representation.

        (goal <g> ^problem-space <p> ^state <s> ^operator undecided)
        (problem-space <p> ^name proof-by-contradiction)
        (state <s> ^ks <ks> -^newirc  -^newprop)
        (ks <ks> ^hint <h2>)
        (hint <h2> ^name insight-hint)
        (operator <o1> ^name elaborate-representation)
        (operator <o2> ^name generate-proposition)
        (preference <o1> ^role operator ^value acceptable ^goal <g>
                ^problem-space <p> ^state <s>)
        (preference <o2> ^role operator ^value acceptable ^goal <g>
                ^problem-space <p> ^state <s>)
-->
        (preference <o1> ^role operator ^value better ^reference <o2> ^goal <g>
                ^problem-space <p> ^state <s>)
)

(sp  prefer-generate-proposition1
;------------------------------------
;; If  new elaborations have been made
;; then try to generate propositions.

        (goal <g> ^problem-space <p> ^state <s> ^operator undecided)
        (problem-space <p> ^name proof-by-contradiction)
        (state <s> ^ks <ks> ^newirc )
        (operator <o1> ^name elaborate-representation)
        (operator <o2> ^name generate-proposition)
        (preference <o1> ^role operator ^value acceptable ^goal <g>
                ^problem-space <p> ^state <s>)
        (preference <o2> ^role operator ^value acceptable ^goal <g>
                ^problem-space <p> ^state <s>)
-->
        (preference <o2> ^role operator ^value better ^reference <o1> ^goal <g>
                ^problem-space <p> ^state <s>)
        (state <s> ^change <change>)
        (delete <change> ^newirc)
)

(sp  prefer-generate-proposition2
;------------------------------------
;; If  new propositions have been recently generated
;; then try to generate some more propositions.

        (goal <g> ^problem-space <p> ^state <s> ^operator undecided)
        (problem-space <p> ^name proof-by-contradiction)
        (state <s> ^ks <ks> ^newprop)
        (operator <o1> ^name elaborate-representation)
        (operator <o2> ^name generate-proposition)
        (preference <o1> ^role operator ^value acceptable ^goal <g>
                ^problem-space <p> ^state <s>)
        (preference <o2> ^role operator ^value acceptable ^goal <g>
                ^problem-space <p> ^state <s>)
-->
        (state <s> ^change <change>)
        (delete <change> ^newprop)
        (preference <o2> ^role operator ^value better ^reference <o1> ^goal <g>
                ^problem-space <p> ^state <s>)
)




;=========================================================================
;;              CONCEPT FORMATION RULES

(sp elaborate-concept-by-analogy
;-------------------------------
;; The hint <h> specifies what concept <ir1> to look at.  The hint also
;; specifies what property of the concept  <focus> to look at.
;; If that property has the value nil, then create a new concept <ir2>
;; that is the same as the old concept in all respects (i.e. analogous)
;; except that the value of nil in the attribute of interest is replaced
;; with a value <val0> found by looking at relevant features
;; of the board <rw1>.  Rather than examine every matching concept (square)
;; on the board, a representative square <ir2> is chosen. (We assume that
;; the subject chooses at least two -- one of each color)

        (goal <g> ^problem-space <p> ^state <s> ^operator <o> )
        (state <s> ^ks <ks> ^rw <rw>)
        (ks <ks> ^hint <h> ^irc <ir1> <ir2>)
        (rw <rw> ^rwe <rw1>)
        (problem-space <p> ^name proof-by-contradiction)
        (operator <o> ^name elaborate-representation)
        (hint <h> ^attend-to-name <name> ^attend-to-attribute <focus>)
        (irc <ir1> ^name <name> ^<focus> nil ^shape <shape> ^number-of <nn>
                ^area <a> ^position <pos> ^function <func> ^status <stat>)
        (irc <ir2> ^name sample ^shape <name> ^number-of <nn> ^position <p1>
                ^salience << 1st 2nd >> )
        (rwe <rw1> ^name <name> ^<focus> {<val0> <> nil} ^position <p1>)
-->
        (operator <o> ^newstateneeded)
        (state <s> ^newirc)
        (ks <ks> ^irc <ir3>)
        (irc <ir3> ^name <val0> <name> concept ^<focus> <val0> ^shape <shape>
                ^number-of <nn> ^area <a> ^position <pos> ^function <func>
                ^status <stat> )

)
;========================================================================
;       RULES FOR CREATING PROPOSITIONS


(sp elaborate-propositions-by-analogy
;-----------------------------
;; The hint <h> specifies which concept <name> and which property <att1>
;; of that concept is of current interest.  Any proposition <pr> that mentions
;; the concept as being covered is chosen to be elaborated. In order to
;; specify which features of the board <rw1>&<rw2> should be used as the
;; basis for the elaboration, a pair of representative-coverees <ir1>
;; which indicate the location of two specific instances <p1>&<p2> of
;; the concept are chosen.  (Note: in this case "concept" is equivalent
;; "square(s)")  The interesting properties of the concept (e.g. color-of
;; the squares) are observed in the two specific instances and then used
;; to elaborate the old proposition.  The new propositions created are
;; exactly the same as (analogous to) the old proposition except that
;; the concepts are now modified according to the observed properties
;; (e.g. "black square" or "white square" instead of simply "square").

        (goal <g> ^problem-space <p> ^state <s> ^operator <o> )
        (problem-space <p> ^name proof-by-contradiction)
        (state <s> ^ks <ks> ^rw <rw>)
        (operator <o> ^name generate-proposition)
        (ks <ks> ^hint <h> ^prop <pr> ^irc <ir1>)
        (rw <rw> ^rwe <rw1> <rw2>)
        (hint <h> ^attend-to-name <name> ^attend-to-attribute <att1>)
        (prop <pr> ^term1 <t1> ^rel covers ^term2 <name> ^term3 <name>)
        (irc <ir1> ^name sample ^shape <name> ^position <p1>
                ^adjacent-to <p2>  ^salience 1st)
        (rwe <rw1> ^shape <name> ^position <p1> ^<att1> <c1>)
        (rwe <rw2> ^shape <name> ^position <p2> ^<att1> <c2>)
-->
        (operator <o> ^newstateneeded)
        (state <s> ^newprop)
        (prop <pr> ^term1 domino ^rel covers ^term2 <c1> square
                ^term3 <c2> square )
        (ks <ks> ^prop <pr>)
)


(sp infer-equal-numbers-covered
;-------------------------------
;; Any coverer that must covers two coverees implies equal numbers of the
;; coverees.

        (goal <g> ^problem-space <p> ^state <s> ^operator <o>)
        (problem-space <p> ^name proof-by-contradiction)
        (state <s> ^ks <ks> ^rw <rw>)
        (operator <o> ^name generate-proposition)
        (ks <ks>  ^prop <pr> ^sf <sf1>)
        (sf <sf1>  ^attend-to-name <item> ^attend-to-attribute number-of)
        (prop <pr> ^term1 <coverer> ^rel covers ^term2 {<c1> <> <item>} <item>
                 ^term3 {<c2> <> <c1> <> <item>} <item>)
-->
        (operator <o> ^newstateneeded)
        (state <s> ^newprop)
        (prop <p2> ^source logical <pr>  ^term1 number <c1> <item> ^rel equal
                 ^term2 number <c2> <item> ^truth true )
        (ks <ks> ^prop <p2>)
)


 (sp make-count-proposition
;-------------------------
;; If the number of two types of squares are not equal, state
;; this fact as an empirical proposition.  (NOTE: Actually in place
;; of this production, future versions of the simulation will
;; probably instantiate elaborate-concept-by-analogy a 2nd time
;; to notice the color of the Xed squares.  Then an inference will
;; be made that there cannot be equal numbers of blacks and whites
;; since two blacks were removed.  Human subjects almost always follow
;; this sequence.)

        (goal <g> ^problem-space <p> ^state <s> ^operator <o>)
        (problem-space <p> ^name proof-by-contradiction)
        (state <s> ^ks <ks> ^rw <rw>)
        (ks <ks> ^hint <h> ^prop <pr> ^irc <ir1> <ir2>)
        (operator <o> ^name generate-proposition)
        (hint <h> ^attend-to-name <item>)
        (prop <pr> -^source empirical ^term1 number {<c1> <> <item>
                <> number} <item>  ^rel <any-relation>
                ^term2 number {<c2> <> <item> <> number} <item>)
        (irc <ir1> ^name number <c1> <item> ^number <n1>)
        (irc <ir2> ^name number <c2> <item> ^number { <n2> <> <n1> })
```

```
-->
        (operator <o> ^newstateneeded)
        (state <s> ^newprop)
        (prop <pr> ^source empirical ^term1 number <c1> <item> ^rel equal
                   ^term2 number <c2> <item> ^truth false )
        (ks <ks> ^prop <pr>)
)

(sp contradiction-found
;-----------------------
;; Check for identical propositions with opposite truth values.
;; Scope of this production is narrowed somewhat by making use of the hint
;; and by restricting examination to those propositions that deal with
;; the number of types of items.
;; Subjects typically exclaim that the problem is impossible as soon as
;; they detect a contradiction.

        (goal <g> ^problem-space <p> ^state <s> ^operator <o> )
        (problem-space <p> ^name proof-by-contradiction)
        (state <s> ^ks <ks> ^rw <rw>)
        (ks <ks> ^hint <h> ^prop <pr1> <pr2> -^contradiction? yes)
        (hint <h> ^attend-to-name <item>)
        (prop <pr1> ^term1 number {<c1> <> <item> <> number} <item> ^rel <rel>
         ^term2 number {<c2> <> <item> <> number} <item> ^truth true)
        (prop <pr2> ^term1 number {<c1> <> <item> <> number} <item> ^rel <rel>
         ^term2 number {<c2> <> <item> <> number} <item> ^truth false)
-->
        (operator <o> ^newstateneeded)
        (ks <ks> ^contradiction? yes)
        (write1  (crlf) | The problem is impossible!|)
)

(sp create-find-reason
;-----------------------------
        (goal <g> ^problem-space <p> ^state <s> ^operator undecided)
        (problem-space <p> ^name proof-by-contradiction)
        (state <s> ^ks <ks> ^rw <rw>)
        (ks <ks> ^contradiction? yes)
-->

        (operator <o1> ^name find-reason)
        (preference <o1> ^role operator ^value acceptable ^goal <g>
                ^problem-space <p> ^state <s>)
)

(sp prefer-find-reason
;-------------------
;; If a contradiction has been detected then it becomes
;; the highest priority to explain it, given that the subject
;; is seraching for a proof by contradiction.

        (goal <g> ^problem-space <p> ^state <s> ^operator undecided)
        (problem-space <p> ^name proof-by-contradiction)
        (state <s> ^ks <ks> ^rw <rw>)
        (ks <ks> ^contradiction? yes)
        (preference <o1> ^role operator ^value acceptable ^goal <g>
                ^problem-space <p> ^state <s>)
        (operator <o1> ^name find-reason)
-->
        (preference <o1> ^role operator ^value best ^goal <g>
                ^problem-space <p> ^state <s>)
)

(sp trace-back-contradiction
;--------------------------
;; Retrieve the contradiction and state the logically-inferred proposition
;; and its premises as support for concluding impossibility since they
;; contradicted by empirical observation.

        (goal <g> ^problem-space <p> ^state <s> ^operator <o>)
        (problem-space <p> ^name proof-by-contradiction)
        (state <s> ^ks <ks> ^rw <rw>)
        (operator <o> ^name find-reason)
        (ks <ks> ^hint <h> ^prop <pr1> <pr2> <pr3>)
        (hint <h> ^attend-to-name <item>)
        (prop <pr1> ^source logical {<pr3> <> logical <> empirical <> nil}
                 ^term1 number {<c1> <> number <> <item>} <item>  ^rel <rel>
                 ^term2 number {<c2> <> number <> <item>} <item>
                 ^truth {<truth1> true})
        (prop <pr2> ^source empirical ^term1 number {<c1> <> number
                <> <item>} <item>  ^rel <rel> ^term2 number
                {<c2> <> number <> <item>} <item> ^truth {<truth2> false})
        (prop <pr3> ^term1 <t>  ^rel <r> ^term2 {<cc1> <> <item>} <item>
                 ^term3 {<cc2> <> <item>} <item>)
-->
        (write1 (crlf) |For the problem to be possible, it must be|
                <truth1> |that|)
        (write1 (crlf) number <c1> <item> <rel> number <c2> <item> )
        (write1 (crlf) |since| <t> <r> <cc1> <item> |and| <cc2> <item> |.|)
        (write1 (crlf) |But, it is| <truth2> |that| )
        (write1 (crlf) number <c1> <item> <rel> number <c2> <item> )
        (write1 (crlf) |by empirical observation.  Therefore, the|)
        (write1 (crlf) |problem is impossible.|)
        (halt)
)

;;============================================================================
;;               David Steier's state copying productions
;;============================================================================
;; Kinds of state modifications needed for operator  implementation:
;;      1) Adding an attribute to the state
;;      2) Deleting an attribute from the state
;;      3) Replacing the value of an existing attribute  on the state
;;              with another value
;;
;; Responses in the current system:
;;      1) create a new state, copy over all the attributes,
;;              and add the new one
;;      2) create a new state, and copy over all the attributes of
;;              the old state except for the one being deleted
;;      3) create a new state and copy over the attributes
;;              of the old state except for the one being replaced,
;;              add the replacing attribute
;;
;; The productions fire when the change augmentations have been
;;      added to the current operator.
;;  "NEWSTATENEEDED" HACK needed so this production will fire once
;; per operator.  I really want to say that a  new state is required
;; if one or more changes are needed.  The augmentation to the operator
;; also lets state adds work correctly.
;; Set up new state and make sure the name is not copied
(sp newstate*set-up-state-for-copying
        (goal <g> ^problem-space <p> ^state <s> ^operator <q>)
        (operator <q> ^newstateneeded)
-->
        (state <ns> ^dummy-attribute dummy)
        (preference <ns> ^goal <g> ^problem-space <p> ^state <s>
                ^operator <q> ^role state ^value acceptable)
)
;; Don't copy the name attribute
(sp newstate*copy-valid-state-attributes
        (goal <g> ^problem-space <p> ^state <s> ^operator <q>)
        (operator <q> ^newstateneeded)
        -{ (operator <q> ^change <change>)
                (delete <change> ^<att> <val>)}
        -{ (operator <q> ^change <change>)
                (replace <change> ^<att> <val>)}
        (state <s> ^{<att> <> name} <val>)
        (preference <ns> ^goal <g> ^problem-space <p> ^state <s>
                ^role state ^operator <q> ^value acceptable)
        (state <ns> -^<att> <val>)
-->
        (state <ns> ^<att> <val>)
)

(sp newstate-do-add-to-newstate-if-needed
        (goal <g> ^problem-space <p> ^operator <q> ^state <s>)
        (operator <q> ^change <a>)
        (add <a> ^<att> <val>)
        (preference <ns> ^goal <g> ^problem-space <p> ^state <s>
                ^operator <q> ^role state ^value acceptable)
        (state <ns> -^<att> <val>)
-->
        (state <ns> ^<att> <val>)
)

(sp newstate-do-replace-to-newstate-if-needed
        (goal <g> ^problem-space <p> ^operator <q> ^state <s>)
        (operator <q> ^change <r>)
        (replace <r> ^{<att> <> by} ^by <newval>)
        (preference <ns> ^goal <g> ^problem-space <p> ^state <s>
                ^role state ^value acceptable)
        (state <ns> -^<att> <newval>)
-->
        (state <ns> ^<att> <newval>)
)
```